Research Paper | Computer Science | E-ISSN No : 2454-9916 | Volume : 8 | Issue : 6 | Jun 2022

NEW EMERGING NETWORK ARCHITECTURE: SOFTWARE DEFINED NETWORK (SDN)

Nishtha

Department of Computer Science, Rajiv Gandhi Govt. Degree College, Chaura Maidan, Shimla, India.

ABSTRACT

One of the latest network architectures is the Software Defined Network (SDN) in which the data and control planes are segregated. This keeps apart, the two important components the packet forwarding and the switches that act as packet carrying devices. The control of forwarding the packet is performed by a programmable software component, the controller, and the forwarding elements such as switches are regulated by the controller. This architecture also provides open interfaces for communication between the controller and the forwarding hardware as well as the controller and network applications. Among a few protocols that exist which provide open interfaces for communication among the controller and forwarding element, OpenFlow (OF) is regarded as a standardized protocol in this architecture. The controller programs the data plane through these open interfaces, whereas, the open interfaces for communication among the controller and the applications allow easy development of network applications. By providing eminence benefits over traditional network architecture in the recent years SDN is one of the important areas of research. Therefore, in this paper, we have highlighted the major functioning of SDN.

KEYWORDS: Software Defined Network, OF, ONF.

INTRODUCTION:

Traditional networks require a number of different protocols to specify a standard in order to meet their technical as well as industrial needs [1, 2]. Such networks are complex [1, 3, 4, 5], vulnerable and static [5, 6], as a result, these networks cannot accommodate changing traffic, have inconsistent policies [7, 8], are difficult to manage, unable to scale, vendor dependent [9, 10, 11, 12, 13, 14] where innovations are almost impossible. Above all, traditional network management requires the manual configuration of each network device [7, 14]. All these drawbacks of traditional networks call for Software Defined Network (SDN) architecture and its associated standards [2, 6, 7, 14].

In traditional networks, the network control, as well as the data plane, lay on the same forwarding element [8, 9, 10]. In Software Defined Network (SDN) architecture, network control is separated from the data plane [8, 11, 12, 13]. The network control in SDN is taken out from the forwarding element and is embedded as software-based logically centralized control known as the controller [6, 8]. This architecture provides open interfaces for communication among the controller and the forwarding element [15] and these may be any of the vendor-independent Application Programming Interfaces (APIs) such as ForCES (Forwarding & Control Element Separation) developed by the Internet Engineering Task Force (IEFT), OpenFlow (OF) developed by ONF (Open Networking Foundation) or any other proprietary protocol [11]. These open interfaces allow the controller to install rules in the forwarding element and accordingly the network traffic moves [15]. SDN also provides Open interfaces for communication between the controller and the applications thereby allowing the development of distinct custom-built applications for network management. As a result, new network capabilities and services may be easily developed as well as deployed in SDN [7, 9].

Section II of the paper introduces the basic concepts of SDN architecture, and the subsequent section III presents a brief on OpenFlow protocol, section IV the control plane, and section V the forwarding elements. The last section VI discusses the conclusion.

Software Defined Networks:

In Software Defined Network architecture, the network control is detached from the data plane [8, 11, 12, 13]. The data plane is developed from special purpose hardware and is only used to forward network traffic [10] whereas the control plane resides in network servers as a logically centralized control program that installs rules in the forwarding element [6, 10, 12]. Traffic in the network is directed in accordance with these rules [1]. While routing, path computation, signaling, etc. are the most common control plane functions, the addition of new features such as virtual private network (VPN), cloud computing, contents distribution networking, fault tolerance, mobility, etc. have resulted in the complexity of the control plane [16].

SDN can conceptually be represented as three-tier architecture as shown in Fig.1.

Infrastructure Layer (Tier-1): Comprises all the forwarding elements and includes software interfaces and hardware components in the forwarding elements.

Control Layer (Tier-2): The control layer regulates and manages forwarding hardware i.e. Tier-1 and consists of a software-based logically centralized controller.

Application Layer (Tier-3): This layer consists of applications and services that make use of the control and infrastructure layer, i.e. Tier-1 and Tier-2. The Open Application Programming Interfaces (APIs) for communication between the controller and applications enable the easy creation of a custom-built network applications [12]. These applications perform all network management tasks and the traffic in the network is moved as per the requirements of these applications [17, 18].

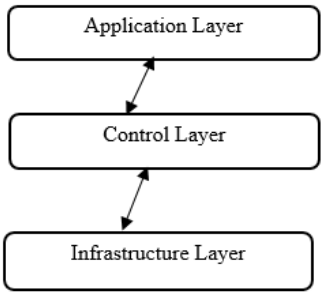

Fig. 1: Software Defined Network: A three-tier architecture

Among many available protocols, OpenFlow (OF) is generally regarded as a de-facto standard for SDNs [10, 13].

OpenFlow (OF):

OpenFlow architecture separates the control and data planes which are then tied by OpenFlow protocol by providing standard interfaces for communication between them [6]. OpenFlow (OF) is an open-source routing southbound protocol for communication between the control and the data plane. By providing a standard programmable interface, OpenFlow permits the programmers to directly control the data plane by providing rules (flows to route packets/flow in the network [6, 10, 12]. Many algorithms for traffic engineering, server load-balancing, data center routing, energy-efficient network management, virtualization, fine-grained access control, traffic monitoring, fault tolerance, denial of service detection, host mobility, etc. using OpenFlow have already been developed by researchers [10], [19].

OpenFlow is a flow-oriented technology and most of the OpenFlow applications send data as flows instead of new functions in switches. DIFANE is a distributed flow management architecture that deals with the traffic in the data plane. In DIFANE, the wildcard rules are distributed among commodity switches, and packets are only forwarded through switches that have necessary rules. Therefore, DIFANE may quickly react to network changes. DevoFlow, scalable architecture is built to overcome the drawbacks of OpenFlow for high-performance networks. It decreases switch-controller interactions by maintaining a reasonable amount of visibility, thus, simplifying the design of high-performance OpenFlow switches.

OpenFlow specifies a flow as a set of header fields, counters, and associated actions. The methodology behind OpenFlow is that the controller first composes network flow rule logic and then specifies the rules in the flow table of one or more OpenFlow-enabled switches dynamically. The flow table already contains a set of flow rules. The incoming packets are matched with these flow rules and accordingly specified action to process the flows in the data plane takes place [12, 18].

Initially, OpenFlow was designed at Stanford University in 2008 [22]. Different versions of the OpenFlow protocol specifications are 1.0.0, 1.1.0, 1.2, 1.3.0, etc. have been developed. 1.0.0, is the widely deployed OF specification. OpenFlow was initially developed for research fields but, presently it is also being used in commercial applications such as data centers [23]. Using OpenFlow for commercial applications raises new challenges such as performance and scalability. In order to provide developers/researchers a clue that how OpenFlow architecture will perform on certain parameters, using a simulation basic model for the forwarding speed and blocking probability of an OF switch with an OF controller is derived that depends on the measurements of switching times of current OpenFlow hardware [23].

One of the important parameters in network management is the measurement of network traffic. Time, as well as accurate measurements, are the two important factors on which network traffic depends. Existing solutions for network measurement such as the use of custom-built hardware or data collection and analysis, are unsuitable because of high overheads. A more accurate, low overhead practical traffic measurement framework is suggested that runs on commodity hardware. In this, the switches match packets against a small collection of rules and update traffic counters for the highest-priority match. A separate controller is used to read the counters which also dynamically adjust the rules and within no time send them to switches on identifying large traffic aggregates. In the initial step towards the development of this framework, a hierarchical heavy hitters (HHH) algorithm is designed and evaluated. This is used to identify large traffic aggregates and to keep the balance between measurement accuracy and switch overhead [24].

There are two methods of communication between the switch and the controller. One is the out-of-band method in which network paths (such as VLANs or separate physical interfaces) that are not affected by OpenFlow operation are required for communication between the two. The other is the in-band control method where the OpenFlow network is used for data transfer as well as communication. The rules are also created in two ways in the OpenFlow networks: proactive and reactive. In proactive style, rules are inserted in switches before they are needed. In reactive style, rules are inserted into switches according to the packets observed by the controller via Packet-In messages. In these networks, when a packet does not match a rule, switches generate a Packet-In message. Such a packet is sent to the controller and the controller takes the appropriate decision to handle the packet.

In 2011, a group of approximately 70 members including researchers, network administrators, network operators, and vendors formed the Open Networking Foundation (ONF) to make OpenFlow the new standard for SDN networks [8, 10, 18].

Controller:
The main logic behind SDNs is that all the switches and the routers move decision-making power to the logically centralized controller which is regarded as a network operating system and enables all control and management-related functions [1].

A set of open APIs allows communication between the controller and switches as well as controller and applications. The interfaces used to communicate between different layers of the SDN stack, as per their functions in SDN architecture, are categorized as:

- Northbound APIs: To communicate between controllers and applications [12, 17].
- Southbound APIs: To communicate between the controller and the underlying hardware.
- East/Westbound APIs: To communicate between groups of controllers for state synchronization [17].

In SDNs, network virtualization performs a significant function. It may be easily implemented in SDN because both the controller applications and switch forwarding tables use APIs. Thus, network virtualization may be added as an extension to the OpenFlow controller. Network virtualization with SDN provides benefits of dynamic resource allocation and also allows each "tenant" with its own network topology and traffic control [23].

Forwarding Elements:
The forwarding elements are switches, routers, and access points which are used to carry network traffic [2, 13]. No special-purpose device or any proprietary hardware is required in an OpenFlow network. Therefore, the devices used can be grouped as dedicated OpenFlow switches, and routers in which traditional Layer 2, and Layer 3 processing (forwarding and routing) are not required. The other type is OpenFlow-enabled general-purpose Ethernet switches, and routers to which Open-Flow protocol and interfaces as new attributes are added. In such devices, Ternary Content Addressable Memory (TCAM) is used to build a flow table.

The data plane of an OpenFlow switch is made programmable by dynamically specifying the rules in a Flow Table of the switch that is logically created by the OpenFlow controller [12]. As a result, flows are routed depending on flow entries in the flow table of switches as displayed in Fig 2.

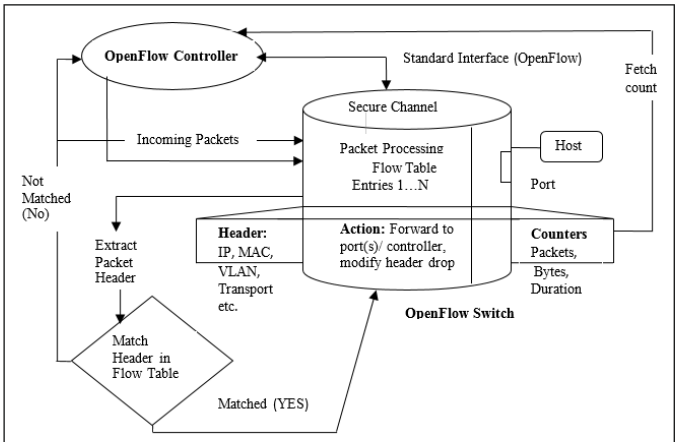

Fig. 2: OpenFlow Architecture for communication between Controller and Switch [23, 25]

Flow Table, OpenFlow protocol, and secure channel are three main components of a switch. Flow Table: It contains a list of flow entries where the incoming packets are compared with the flow entries. Each flow entry contains (i) a packet header, (ii) action, and (iii) statistics. Packet Header defines the flow rules. Incoming packets are compared with the header field of each flow entry and if matched, the packets are processed as per the action specified in that entry. For retaining the flow statistics Counters are used [9]. Therefore, as a packet comes, a highest-priority matching rule is selected by the switch, counters are updated, and the specified action is executed. In case no rule is matched, the switch sends the packet header to the controller and waits for a response from the controller.

The OpenFlow Protocol: It provides interfaces that allow the controller to change flow rules in the flow table of a switch [23, 25]. In this way, the rules are specified in the switches as per the authorization of the programmable centralized controller. Consequently, this enables us to list countless benefits of SDN.

CONCLUSION:
The main reasons for Software Defined Network architecture to gain importance are: (a) separation of control planes and (b) logically centralized software control program that is taken out from the network and provides a global network view have resulted in a marked difference between SDN and traditional network and thereby increases the importance of SDN.

As a result, the controller abstracts the underneath forwarding hardware from network applications and this ultimately culminates in a number of benefits ranging from easy innovations, high scalability, simplified network management, robust fault tolerance, and infrastructure saving to many more. Thus, SDN emergence has resulted in reconsidering our viewpoints regarding network architecture and design.

REFERENCES:
I. Peyman Kazemian, George Varghese, Nick McKeown. (2012). Header Space Analysis: Static Checking For Networks, Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation NSDI'12, USENIX, pp 9-9.

II. Mark Reitblatt, Marco Canini, Arjun Guha, Nate Foster. (2013). FatTire: declarative fault tolerance for software-defined networks, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking HotSDN'13, ACM, pp 109-114.

III. Nitheesh Murugan Kaliyamurthy, Swapnesh Taterh, Suresh Shanmugasundaram, Ankit Saxena, Omar Cheikhrouhou. (2021) Software-Defined Networks, Security and Communication Networks, Hindawi.

IV. Brandon Heller, Colin Scott, Nick McKeown, Scott Shenker, Andreas Wundsa, Hongyi Zeng, Sam Whitlock, Vimalkumar Jeyakumar, Handigol, James McCauley, Kyriakos Zarifis, Peyman Kazemian. (2013) Leveraging SDN Layering to Systematically Troubleshoot Networks, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking HotSDN'13, ACM, pp 37-42, doi= 10.1145/2491185.2491197.

V. Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, Hui Zhang. (2005) A clean slate 4D approach to network control and management, Newsletter ACM SIGCOMM

Computer Communication Review, Volume 35 Issue 5, ACM, doi = 10.1145/1096536.1096541. 41-45.

VI. Arjun Guha, Mark Reitblatt, Nate Foster, Machine-Verified Network Controllers (2013), Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation PLDI'13, ACM, pp 483-494, doi=10.1145/2491956.2462178, 2013.

VII. Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Towards Secure and Dependable Software-Defined Networks, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking HotSDN'13, ACM, pp 55-60, doi= 10.1145/2491185.2491199, 2013.

VIII. Seyed Kaveh Fayazbakhsh, Vyas Sekar, Minlan Yu, Jeffrey C. Mogul, FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions, Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking HotSDN '13, ACM, pp 19-24, doi = 10.1145/249118-5.2491203, 2013.

IX. Nate Foster, Michael J. Freedman, Arjun Guha, Rob Harrison, Naga Praveen Katta, Christopher Monsanto, Joshua Reich, Mark Reitblatt, Jennifer Rexford, Cole Schlesinger, Alec Story, and David Walker, Languages for software-defined networks, IEEE (Volume:51, Issue: 2) Communications Magazine, pp 128–134, doi = 10.1109/MCOM.2013.6461197, 2013.

X. Christopher Monsanto, Nate Foster, Rob Harrison, David Walker, A Compiler and Run-time System for Network Programming Languages, Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL'12, ACM, pp 217-230, doi= 10.1145/2103656.2103685, 2012.

XI. M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. A. Correa, S. C. de Lucena, and M. F. Magalhaes, Virtual routers as a service: the RouteFlow approach leveraging software-defined networks, Proceedings of the 6th International Conference on Future Internet Technologies CFI '11, ACM, pp 34-37, doi =10.1145/2002396.2002405, 2011.

XII. Seugwon Shin, Phillip Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, Mabry Tyson, FRESCO: Modular Composable Security Services for Software-Defined Networks, NDSS Symposium 2013, 2013.

XIII. Jon Matias, Borja Tornero, Alaitz Mendiola, Eduardo Jacob, Nerea Toledo, Quality of Transmission Awareness in Converged Electronic and Optical Networks with OpenFlow, 2012 European Workshop on Software Defined Networking,2012

XIV. Stefano Vissicchio, Laurent Vanbever, Olivier Bonaventure, Opportunities and Research Challenges of Hybrid Software Defined Network, In ACM Computer Communication Review (Editorial Zone), 44(2), 2014.

XV. Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker, Abstractions for network update, Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication SIGCOMM '12, ACM, pp 323-334, doi = 10.1145/2342356.2342427, 2012.

XVI. Kwangtae Jeong, Jinwook Kim, Young-Tak Kim, QOS-aware Network Operating System for Software Defined Networking with Generalized OpenFlows, In the Network Operations and Management Symposium (NOMS), 2012 IEEE, April 2012, pp. 1167 - 1174.

XVII. Manu Sood, Nishtha, Traditional verses Software Defined Networks: A review paper, Published in IJCEA, 2014.

XVIII. Open Networking Foundation (ONF): https://www.opennetworking.org.

XIX. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, OpenFlow: enabling innovation in campus networks, Computer Communication Review Newsletter, ACM SIGCOMM, Volume 38, Issue 2, pp 69-74, ACM, doi = 10.1145/1355734.1355746, 2008.

XX. Minlan Yu, Jennifer Rexford, Michael J. Freedman, Jia Wang. Scalable flow-based networking with DIFANE, Proceedings of the ACM SIGCOMM 2010 conference SIGCOMM'10, ACM, pp 351-362, doi = 10.1145/1851182.1851224, 2010.

XXI. Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, DevoFlow: scaling flow management for high-performance networks, In Proceedings of the ACM SIGCOMM conference SIGCOMM '11, ACM, pp 254-265, doi = 10.1145/2018436.2018466,2011.

XXII. Jon Matias, Borja Tornero, Alaitz Mendiola, Eduardo Jacob, Nerea Toledo, Implementing Layer 2 Network Virtualization Using OpenFlow: Challenges and Solutions, European Workshop on Software Defined Networking (EWSDN), 2012, pp 30 - 35, doi = 10.1109/EWSDN.2012.18, 2012.

XXIII. Michael Jarschel, Simon Oechsner, Daniel Schlosser, Rastin Pries, Sebastian Goll, Phuoc Tran-Gia, Modeling and performance evaluation of an OpenFlow architecture, Proceedings of the 23rd International Teletraffic Congress ITC '11, pp 1-7, 2011.

XXIV. Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, Sujata Banerjee. DevoFlow: scaling flow management for high-performance networks, In Proceedings of the ACM SIGCOMM conference SIGCOMM '11, ACM, pp 254-265, doi = 10.1145/2018436.2018466,2011.

XXV. Sajad Shirali-Shahreza, Yashar Ganjali. FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow. (2013), Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking HotSDN'13, ACM, pp 167-168, doi= 10.1145/2491185.2491215.